# 简介

## 目的

本文档主要用于介绍FTL设计思路，针对的人群：

1. SSD、Emmc、UFS固件研发人员
2. 管理者
3. 对FTL感兴趣的小伙伴

## 范围

本文档基于以下要求设计的：

1. 需实现LogWrite、LogRead、TabManage、GC、FDL五个模块。
2. 基于硅格自研的仿真平台、新建一个新的FTL工程、工程命名为eMMC\_FW\_TEST。
3. Flash Mode可以选择2D MLC L83A做为仿真调试的Flash模型（Datasheet请参考kb.com），仿真采用1Ch1CE方式（1颗L83Aflash）、采用直接MLC编程的方式。
4. FW能通过仿真64K满盘大数据顺序写、读校验TestCase两次测试。

注意：本设计暂时未涉及上电恢复、坏块管理、磨损均衡以及正常/异常掉电处理，但预留了部分功能接口。目前只涉及映射表管理、垃圾回收两大功能模块。

背景及设想

本设计以Micron的L83A（4G）为例，具体参数page：8K+744Byte;block：256page；plane：1064blocks，lun=2planes=2128blks，设计读写方式全采用MLC方式。

假设块0,1作为存FW，块2存放一级索引及writepoint等信息，块3….31做sysblk存放固件产生的表格数据（如二级映射表，PS表，VPC表、PEC表），后面2096个块作为Data blk，用于存放用户数据。

假设总的用户数据为2048\*256\*8K。总的logic\_sector\_max:2048\*256\*(8K/512)= 8388608。采用physical address为U32的，映射表太大，本设计采用2级映射方式。512K/2K=(256)。256\*4B=1024B。在内存中开辟1048Byte 的一级映射空间。

设计L83A块功能分区图1：



图1.块功能分区图

缩写定义说明如表1

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Description** | **Glossary or Information** |
| Bad Block | Bad Block | 一个无法用于正常数据存储的坏块 |
| Cold Block | Cold Block | 一些拥有很久前写入数据的块 |
| Data Block | Data Block | 一个至少拥有一个有效数据页的块 |
| DWL | Dynamic wear leveling | 一种平衡磨损机制,但它只考虑为空白的平均磨损 |
| PEC | Program/Erase Count | 擦除次数 |
| Free Block | Free Block | 里面全是无效数据的块 |
| GC | Garbage Collection | 一种将垃圾数据空间回收的机制 |
| LPA | Logical Page Address | 逻辑页地址 |
| PPA | Physical Page Address | 物理页地址 |
| Sector | Sector | Host最基本的读写单元. |
| SWL | Static wear leveling | 一种能让磨损均衡作用于所有块的机制 |
| VPC | Valid Page Counter | 有效页的数量 |
| WA | Write Amplification | 写入放大 |
| WL | Wear Leveling | 一种让所有块能较平衡磨损的机制 |
| WRpoint | Write point | 数据写入地址，一般用于指向下一个写入点 |
| PS | Page Status | 页状态 |

表1 缩写说明表

# 功能模块

## 介绍

根据需求，本设计功包括以下几个模块：

1. 逻辑读写模块
2. 映射表管理模块
3. 垃圾回收模块
4. FDL驱动模块

上层模块结构框图2：



图2上层模块结构框图

## 逻辑读写模块

功能：支持host端读写操作。

根据对sim studio平台相关资料的研读，Host端与emmc端数据传输流程图如图3：



图3 Host端与emmc端数据传输流程图

设计的逻辑写流程图如图4：



图4.逻辑写流程

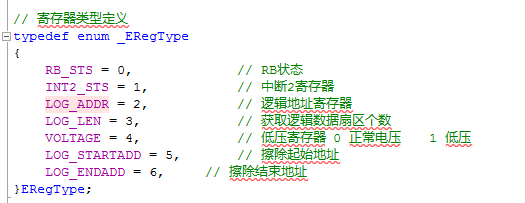
设计的逻辑读流程图如图5：



图5.逻辑读流程

### 数据结构

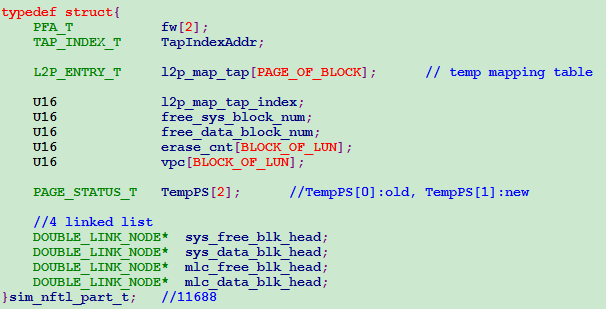
1. 寄存器接口地址：

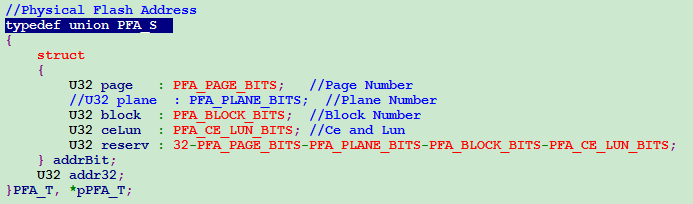


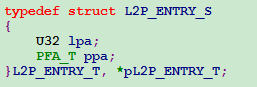
寄存器值含义：

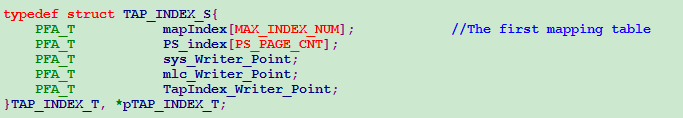


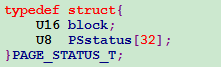
数据结构：











### 函数接口

1.逻辑写函数



2.逻辑读函数



3.获取映射关系函数



4.更新临时映射表函数



5.更新Vpc，临时PS表函数



6.往NAND flash写临时映射表



7. 往NAND flash写临时PS表



8.往NAND Flash写一级映射表，PS表的一级映射、WRpoint



9.Sys Blk 垃圾回收函数



## 映射表管理模块

映射可分为两种：逻辑地址到物理地址映射，称为为直接映射（L2P）；物理地址到逻辑地址映射，称为逆映射（P2L）。逆映射一般用于特殊情况映射关系的恢复，比如说操作过程中断电，或某张映射表格所在的物理页坏，需要重建等。

本设计只利用直接映射（L2P）来管理映射表。由于SRAM资源有限，把所有映射关系存放在Flash中，FW只会加载正使用的那段映射关系（L2P）到内存中。

根据前面分析，本设计采用二级映射，在RAM中开辟1048Byte的一级映射表U32 mapIndex[262]，1个临时映射 。其中**一**级映射表用来索引二级映射表，二级映射表存放的是data写入到物理地址，整体框架如图6：



图6 映射关系整体框架设计图

写过程中，RAM会频繁从Flash中读取和更新临时二级映射表，整个流程图如图7。



图7 写过程中更新临时二级映射表

更新映射表流程如图8：



图8 更新临时映射表流程

### 数据结构

### 函数接口

1.更新临时映射表函数



2.往NAND flash写临时映射表，更新一级映射表





3.从NAND flash 读取MAP表信息



## 垃圾回收模块

垃圾回收（garbage collection）主要分为2种：主动GC（也称闲时GC），和被动GC（用户数据在进行交互的时候，由于没有多余的块而进行的垃圾回收）。本设计只考虑被动GC。

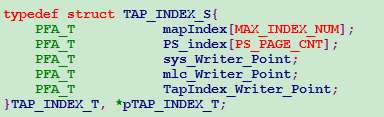
考虑节约内存开销，建立4个链表，SysFreeBlkList，SysDataBlkList，MlcFreeBlkList，MlcDataBlkList。分别存放相应的blk号。

垃圾回收又包括system block垃圾回收和DATA block垃圾回收，大致流程如图9。



图9 整体GC过程

1.块2存放一级索引及writepoint等信息：具体内容为



所占大小为1096Byte，本次设计其每次写入一个新页，当块2填满时，只有最后写入的页内容是有效的，其垃圾回收过程非常简单：把整页可读取到内存或者另外一个块的某位置，擦除块2，然后再搬移最后写入页到块2。

2.块3~31，sys blk，当MLC Data blk写满时会刷写一次临时映射表、临时PS表、PEC表、VPC表（因为PEC和VPC表每次都会全部更新，写入PEC和VPC时，不计VPC）。每次写入顺序如下

|  |
| --- |
| 映射表 |
| PS表信息 |
| PS表信息 |
| PEC表信息 |
| VPC表信息 |

Sys Blk GC设计流程图如图10：



图10 Sys Blk GC过程

3.MLC Blk GC大框架如图11，详细设计流程如图12：



图11 MLC Blk GC整体框架图

图12 Mlc Blk GC详细设计过程

### 数据结构

### 函数接口

1．Sys blk GC函数接口



1. MLC blkGC函数





3.查找VPC最小的blk函数



## FDL(NAND Flash 驱动层)模块

根据研读sim studio平台相关资料，理解NAND Flash驱动层代码，其实就是构建命令描述符。往命令描述符里填充相应数据。

### 数据结构

### 函数接口

1．Flash擦除函数接口



2.Flash 写函数接口



1. Flash写UseData函数



1. Flash 读函数接口



1. Flash读并校验UseData函数



# 附录

1. FW代码具体参见压缩包
2. 配置文件见压缩包

3.64K满盘大数据顺序读写测试代码：

